PATENT APPLICATION

Title: SYSTEMS AND METHODS OF PARTITIONING DATA TO
            FACILITATE ERROR CORRECTION

Inventor(s):   Theodore Carter Briggs
            1316 Swiss Court
            Plano, TX 75023

            Jay Tsao
            4515 Risinghill Drive
            Plano, TX 75024

TITLE: SYSTEMS AND METHODS OF PARTITIONING DATA TO
            FACILITATE ERROR CORRECTION

CROSS REFERENCE TO RELATED APPLICATIONS

The present application is related to Tsao, et al. U.S. Patent Application
entitled "Systems and Methods of Routing Data to Facilitate Error Correction", Filed
July 29, 2003, Attorney Docket No. 200312693-1, which is assigned to the same
assignee as the present application and which is incorporated herein by reference.

TECHNICAL FIELD

The present invention relates to error correction and, more particularly, to the
partitioning of data to facilitate error correction.

BACKGROUND OF INVENTION

Error codes are commonly used in electronic systems to detect and/or correct
data errors, such as transmission errors or storage errors. One common use of error
codes is to detect and correct errors with data stored in a memory of computer system.
For example, error correction bits, or check bits can be generated for data prior to
storing data to one or more memory devices. The error or correction bits are
appended to the data to provide a data structure that is stored in memory. When the
data is read from the one or more memory devices, the check bits can be used to
detect or correct errors within the data. Errors can be introduced, for example, either
due to faulty components or noise in the computer system. Faulty components can
include faulty memory devices or faulty data paths between the devices within the
computer system, such as faulty pins.

Error management techniques have been developed to mitigate the effects
associated with these errors. One simple technique used for personal computers is
known as parity checking. Parity checking utilizes a single bit associated with a piece
of data to determine whether there is a single bit error in the data. Parity checking
cannot detect multiple bit errors and provided no means for correcting errors. A more
sophisticated system, such as a server, uses error correction codes (ECCs) to detect
and correct some errors. An error correction code (ECC) consists of a group of bits,
or codes, associated with a piece of data. A typical ECC system may use eight ECC
bits (check bits, correction bits) for a 64-bit piece of data. The ECC bits provide
enough information for an ECC algorithm to detect and correct a single bit error, or to
detect double bit errors.

One error correction feature employed by servers is referred to in the industry as chipkill. The term chipkill refers to the ability to correct multiple bit errors in memory, where multiple bit errors are based on the width of the memory device. For example, for a 32Mbit dynamic random access memory (DRAM) device that is 4 bits

5   wide, a system that supports a chipkill function would be able to correct a 4-bit wide error in the memory device. Thus, the failure of an entire DRAM chip during a DRAM cycle (*e.g.*, read operation, write operation) organized into a 4-bit width configuration that supports chipkill would not cause the system to fail. Chipkill allows a system to operate in the event of multiple bit errors in any one memory

10   device.

SUMMARY OF INVENTION

The following presents a simplified summary of the invention in order to provide a basic understanding of some aspects of the invention. This summary is not

15   an extensive overview of the invention. It is intended to neither identify key or critical elements of the invention nor delineate the scope of the invention. Its sole purpose is to present some general concepts of the invention in a simplified form as a prelude to the more detailed description that is presented later.

The present invention relates to systems and methods for detecting and

20   correcting errors in data structures. The systems and methods employ error correction code (ECC) techniques that detect and correct errors in a data structure. The data structure is partitioned into separate adjacent bit pair domains, such that a single adjacent bit pair from each memory device is assigned to a given domain. Data associated with a given adjacent bit pair domain can include data bits and check bits

25   that are employed by an ECC technique to detect and correct data bit errors (*e.g.*, single bit errors, adjacent double bit errors) associated with the adjacent bit pair domain.

In one aspect of the invention, systems and methods are provided for detecting and correcting errors in a data structure, for example, during a read or write operation

30   of a system memory. The data structure is stored in a plurality of memory devices associated with a memory address. A data separator partitions the data structure into adjacent bit pair domains, such that a single adjacent bit pair from each of the plurality of memory devices is assigned to an adjacent bit pair domain. An error

detection and correction (EDC) component detects and corrects errors associated with a given adjacent bit pair domain.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 illustrates a block diagram of a system for detecting and correcting bit errors in a data structure in accordance with an embodiment of the present invention.

FIG. 2 illustrates a block diagram of an adjacent bit pair domain configuration in accordance with another embodiment of the present invention.

FIG. 3 illustrates a block diagram of one particular implementation of a system for detecting and correcting bit errors associated with a data structure in accordance with an embodiment of the present invention.

FIG. 4 illustrates a block diagram of an alternate adjacent bit pair domain configuration in accordance with an embodiment of the present invention.

FIG. 5 illustrates a block diagram of a memory system with parallel error detection and correction in accordance with an embodiment of the present invention.

FIG. 6 is a flow diagram illustrating a methodology for storing a data block in accordance with an embodiment of the present invention.

FIG. 7 is a flow diagram illustrating a methodology for reading a data structure in accordance with an embodiment of the present invention.

DETAILED DESCRIPTION

The present invention relates generally to systems and methods for detecting and correcting bit errors in data structures. The systems and methods employ error correction code (ECC) techniques that detect and correct single bit errors and adjacent bit pair errors in a data structure. The systems and methods are operative to process data structures with more bits than can be detected and corrected by the ECC techniques employed. This is accomplished by partitioning a data block and/or data structure into separate domains equal to the number of bits that can be processed by the ECC technique. Chipkill is achieved by populating the separate domains with adjacent bit pairs, such a single adjacent bit pair from each memory device is assigned to a given adjacent bit pair domain.

The data bits in the separate adjacent bit pair domains are processed by one or more error correction units sequentially or in parallel and then recombined to be

written into memory or transmitted to a requestor (*e.g., via* a crossbar device). As used herein, a data block is defined as data that is transmitted or received from a crossbar device, while a data structure is the data and check bits or correction bits associated with the data block that are stored in memory corresponding to a given memory address. The terms data block and data structure are utilized for clarity and not meant to define any specific limiting data format.

FIG. 1 illustrates a system 10 for detecting and correcting errors in a data structure in accordance with an aspect of the present invention. The system 10 can be a server or other computer system that performs error correction associated with a system memory. The system 10 includes an error correction unit 12, a data separator/combiner 14 and a system memory 16. The system memory 16 is comprised of a plurality of memory devices 18, labeled memory device #1 through #K, where K is an integer greater than one. The memory devices 18 can be for example, but not limited to, single-in-line memory modules (SIMM), dual-in-line memory modules (DIMM) and dynamic random access memory (DRAM) modules or other types of memory devices. The system memory 16 is coupled to a data bus 24 and an address bus 26. The size of the data bus 24 and the data structure stored and read into memory in a single memory cycle is equal to the number of memory devices multiplied by the bit column width of the memory devices.

For example, if the system memory is comprised of 72 memory devices having 4-bit width columns, then the size of the data bus 24 and data structure stored and read in a single memory cycle would be 288 bits. However, an ECC checker and corrector for a 288 bit data structure would be impractical. Therefore, the present invention partitions the data structure into separate adjacent bit pair domains comprised of adjacent bit pairs from each memory devices, such that a single adjacent bit pair from each of the memory devices is assigned to a given domain. Error detection and correction can then be performed on data bits associated with the separate adjacent bit pair domains sequentially or in parallel.

The error correction unit 12 is coupled to a crossbar device (not shown), for example, over a crossbar bus 20. The crossbar device can be coupled to a plurality of microprocessor devices, input/output devices and/or memory devices (*e.g.,* one or more cache memory devices). The crossbar device is a switching/ routing system that allows for communications between a plurality of different devices. The error

correction unit 12 can be part of a memory controller or cache coherency controller. The functionality of the error correction unit 12 can be comprised of hardware, software or a combination of hardware and software. It is to be appreciated that the error correction unit 12 can be formed of at least a portion of an application specific integrated circuit or very large scale integrated (VLSI) circuit.

The error correction unit 12 includes an error corrector 13 and a check bit generator 15. The error correction unit 12 is operative to receive data blocks from the crossbar device, partition the data blocks into a plurality of adjacent bit pair domain data sets, and generate check bits for each of the plurality of adjacent bit pair domains. The adjacent bit pair domains are populated with the check bits and data bits from the data block. The adjacent bit pair domains are assigned adjacent data bit pairs per memory device as discussed above. The number of adjacent bit pair domains is based on the column width of the system memory 16. For example, two adjacent bit pair domains are employed for memory devices with a 4-bit column width, while four adjacent bit pair domains would be employed for memory devices with an 8-bit column width.

The adjacent bit pair domains are transmitted over a mid-bus 22 to the data separator/combiner device 14 (*e.g.*, buffer/multiplexer device). The data separator/combiner 14 can be a single integrated device or a plurality of devices that perform the functionality associated with the data separator/combiner 14. The adjacent bit pair domains are transmitted in a sequential order and buffered by the data separator/combiner 14. Once all of the data associated with the adjacent bit pair domains are received by the data separator/combiner 14, the adjacent bit pair domain data is combined into a complete data structure and transmitted over a data bus 24 to be stored in the system memory 16 based on the selected address of the address bus 26.

For example, during a read operation, a data structure associated with an address is read from the plurality of memory devices 18 and provided to the data separator/combiner 14 *via* the data bus 24. For example, if each memory device has a 4-bit column width, 4 bits from each memory device are provided to the data separator/combiner 14. If each memory device has an 8-bit column width, 8 bits from each memory device 18 are provided to the data separator/combiner 14. The data separator/combiner 14 partitions the data structure into adjacent bit pair domains.

Each adjacent bit pair domain includes a data portion and a correction bit portion. The number of bits in an adjacent bit pair domain corresponds to the number of bits correctable by the error corrector 13 in the error correction unit 12.

5   Each adjacent bit pair domain is assigned adjacent bit pairs associated with the plurality of memory devices corresponding to a given address, such that a single adjacent bit pair from each memory device is assigned to a given adjacent bit pair domain, and no more than one adjacent bit pair from a single memory device is assigned to a given domain. The data bits associated with a first adjacent bit pair domain are multiplexed *via* the data separator/combiner 14 and transmitted over the

10   mid-bus 22 to be processed by the error corrector 13. It is to be appreciated that the data bits associated with a given domain can be further partitioned and transmitted sequentially over the mid-bus 22 and recombined at the error correction unit 12 prior to error correction by the error corrector 13.

       While the data bits from the first adjacent bit pair domain are being processed

15   by the error corrector 13, the data bits or portions of the data bits from the second adjacent bit pair domain are transmitted over the mid-bus 22 to the error corrector 13, such that a pipeline process is established. Subsequent data bits from additional domains, if applicable, are transmitted over the mid-bus 22 and processed by the error corrector 13, until all of the data bits associated with the data structure have been

20   processed by the error corrector 13. Once all of the data bits have been checked and corrected for single bit errors and adjacent double bit errors, the check bits are discarded, the data associated with the adjacent bit pair domains are reorganized into a data block and transmitted to the crossbar device and routed to the initial requestor (*e.g.*, a processor, I/O device).

25   It is to be appreciated that the error correction unit 12 can include additional correctors, such that correction can be performed in parallel on data bits associated with different domains to facilitate speed associated with error correction. Additionally, the error correction unit 12 can include additional check bit generators, such that check bits can be assigned and appended to data bits associated with

30   different domains to facilitate speed associated with check bit generation.

       FIG. 2 illustrates an adjacent bit pair domain configuration 40 in accordance with an aspect of the present invention. The adjacent bit pair domain configuration 40 provides chipkill functionality to system memory devices (*e.g.*, for servers) associated with ECC techniques that can correct single and adjacent double bit errors with

memory data buses larger than the capabilities of ECC techniques. The domain configuration illustrates a plurality of memory devices 42, labeled memory device #1 through K, where K is an integer greater than 1. Each of the memory devices 42 has a row associated with a given row address illustrated in FIG. 2 as row address A.

5   During reading and writing of row address A, data bits associated with the row corresponding to row A are concurrently provided at the system memory data bus. Each row of the memory devices 42 has a column width N, where N is an integer multiple of 4 (*e.g.*, 4, 8, 16, 32, etc.)

A first adjacent bit pair from each memory device 42, labeled bit #1 and bit

10   #0, is assigned to adjacent bit pair domain 0. A second adjacent bit pair from each memory device 42, labeled bit #3 and bit #2, is assigned to adjacent bit pair domain 1. Subsequent adjacent bit pairs from each memory device 42 are assigned to subsequent adjacent bit pair domains, while the last set of bits labeled #N-1 and #N-2 from each memory device 42 are assigned to adjacent bit pair domain M-1, where M is equal to

15   N/2. FIG. 2 illustrates the interleaving of adjacent bit pairs to corresponding adjacent bit pair domains, such that one adjacent bit pair from each memory device is assigned to a given domain during a memory read or write cycle. The above adjacent bit pair domain configuration can be scaled up or down depending on the number of bits in the adjacent bit pair domains, the bit column width of the memory devices in the

20   system memory, and the number of bits that can be processed by the error correction technique being employed.

FIG. 3 illustrates one particular implementation of a system 60 for detecting and correcting data errors in accordance with an aspect of the present invention. The system 60 includes a cache coherency controller 62, a buffer/multiplexer 64 and a

25   system memory 66. The cache coherency controller 60 includes functionality associated with performing error detection and correction for single bit errors and adjacent double bit errors associated with a data structure stored in the system memory 66. The cache coherency controller 62 also includes functionality associated with check bit generation to provide check bits employed during error detection and

30   correction. It is also to be appreciated that the cache coherency controller 62 can include other functionality associated with controlling access to a crossbar device (not shown) over a crossbar bus 76.

The system memory 66 is comprised of a plurality of DRAM devices 68, labeled memory device #1 through #K, where K is an integer greater than one. For

illustrative purposes, the DRAM devices 68 of FIG. 3 will be discussed as being 4-bit column width devices. However, other column width devices (*e.g.,* 8, 16, 32, 64, etc.) can be employed in accordance with the present invention. The system memory 66 is coupled to a data bus 80 and an address bus 82. The size of the data bus 80 and data

5   structure read and written to the system memory during a DRAM cycle is equal to the number of memory devices 68 multiplied by the bit column width of the memory devices 68. In the present example, the system memory 66 is comprised of 72 DRAM devices with 4 bit-width columns. Therefore, the size of the data bus 80 and data structure stored and read in a single memory cycle is 288 bits.

10   The cache coherency controller 62 includes an error detection and correction (EDC) component 72 that performs error correction and detection on 144 bits including 12 check bits (*e.g.,* 132 data bits, 12 check bits), which is a one of two portions of the 288 bit data structure stored and read in a single memory cycle. The EDC component 72 is operative to detect and correct both single bit errors and

15   adjacent double bit errors. The functionality of the EDC component 72 can be comprised of hardware, software or a combination of hardware and software. The cache coherency controller 62 and the EDC component 72 can be formed from an application specific integrated circuit or very large scale integrated circuit (VLSI). Although the EDC component 72 is illustrated as residing in a cache coherency

20   controller 62, it is to be appreciated that the EDC component 72 can reside in any memory controller or in a stand-alone component.

        The cache coherency controller 62 is coupled to a crossbar device (not shown) via the crossbar bus 76 and a mid-bus 78. The crossbar device can be coupled to a plurality of microprocessor devices, input/output devices and/or memory devices

25   (*e.g.,* one or more cache memory devices), and allows for communications and routing between the various devices of the system 60. The mid-bus 78 connects the cache coherency controller 62 to the buffer/multiplexer device 64. The buffer/multiplexer device 64 is coupled to the system memory 66 *via* the data bus 80.

        During a read operation, a data structure associated with an address is read

30   from the plurality of DRAM devices 68 and provided to the buffer/multiplexer 64 *via* the data bus 80. The buffer/multiplexer 64 separates or partitions the 288 bit data structure into a first adjacent bit pair domain of 144 bits and a second adjacent bit pair domain of 144 bits. The first adjacent bit pair domain includes adjacent bit pairs from the first and second columns of the DRAM devices 68 associated with a given row

8

address, and the second adjacent bit pair domain includes adjacent bit pairs from the third and fourth columns of the DRAM devices 68 associated with the given row address.

The data bits associated with the first adjacent bit pair domain are further partitioned into a first 72-bit portion and a second 72 bit portion. The first 72-bit portion and the second 72-bit portion associated with the first adjacent bit pair domain are transmitted over the mid-bus 78 sequentially to a reorder component 70. The reorder component 70 then reorders the first and second 72-bit portion into the original 144 bits associated with the first adjacent bit pair domain. The 144 bits associated with the first adjacent bit pair domain are then provided to the EDC component 72. The buffer/multiplexer 64 partitions the 144 bits associated with the second adjacent bit pair domain into a first 72-bit portion and a second 72-bit portion, which are sequentially transmitted to the reorder component 70, as the EDC component 72 is processing the 144 bits associated with the first adjacent bit pair domain. This process is repeated for subsequently read adjacent bit pair domains, such that a processing pipeline can be established. In one aspect of the invention, the data bus 80 operates at 125 MHZ and the mid-bus 78 operates at 500 MHZ.

The EDC component 72 separates the 132 data bits and the 12 check bits from the 144 bits associated with the respective adjacent bit pair domain. A syndrome is computed using the 132 data bits to produce a 144 bit error mask. The error mask is exclusive-OR'ed with the original 144 bit of data and check bits. The single bit errors and adjacent double bit errors are corrected. The 12 check bits are discarded and the remaining 132 bits are provided to a data organizer 74. The data organizer 74 reorganizes the 132 bits into 128 bit data blocks to be transmitted to the crossbar device to a requestor (*e.g.,* a processor, I/O device).

During a write operation, a data block to be written to system memory 66 is received by the data organizer 74. The data organizer 74 assigns or separates adjacent data bit pairs to separate adjacent bit pair domains based on the column widths of the DRAM devices 68. The data bits are then provided to the reorder component 70 and a check bit generator 75. The check bit generator generates 12 check bits or correction bits associated with each adjacent bit pair domain. The check bits are then provided to the reorder component 70 to populated the adjacent bit pair domain with its associated check bits. Once all of the adjacent bit pair domains have been populated with data bits and check bits, the adjacent bit pair domains are transmitted

over the mid-bus 78 to the buffer/multiplexer 64. The buffer/multiplexer 64 combines the data associated with the adjacent bit pair domains to form a data structure, which is then stored in the system memory 66 based on a selected address of the address bus 82.

5   FIG. 4 illustrates an alternate adjacent bit pair domain configuration 100 in accordance with an aspect of the present invention. The adjacent bit pair domain configuration 100 provides chipkill functionality for the system memory 66 of FIG. 3 that employs an ECC technique that can correct single bit errors and adjacent double bit errors for 144-bits of data with a memory data bus that is 288 bits wide. The

10   configuration illustrates a data structure 102 associated with a DRAM cycle (*e.g.,* read cycle, write cycle) having B data bits, where B is an integer multiple of 4 that is greater than or equal to 8. The data structure 102 includes a plurality of 4-bit wide nibbles 104 each being associated with a respective DRAM device.

            A first adjacent bit pair of each of the 4-bit wide nibbles 104 are assigned to a

15   first adjacent bit pair domain 106 (Domain 0) and the second adjacent bit pair of each of the 4-bit wide nibbles are assigned to a second adjacent bit pair domain 108 (Domain 1). The first adjacent bit pair domain 106 includes bits 0 through S-1, while the second adjacent bit pair domain 108 includes bits 0 through R-1, where $R=S=B/2$. The bits associated with the first adjacent bit pair domain 106 can be processed by an

20   EDC component sequentially or in parallel with the bits associated with the second adjacent bit pair domain 108, such that chipkill for the memory system 66 is achieved.

            FIG. 5 illustrates a memory system 120 with parallel error detection and correction in accordance with an aspect of the present invention. The system 120 includes a memory controller 122, a data separator/combiner 124 and a system

25   memory 126. The memory controller 122 includes a first EDC component 132 for performing error detection and correction on bits associated with a first adjacent bit pair domain, and a second EDC component 134 for performing error detection and correction on bits associated with a second adjacent bit pair domain. This provides for faster processing of data associated with read operations of data structures from

30   system memory than systems employing a single EDC component. It is to be appreciated that the memory controller 122 can also include multiple check bit generators (not shown) to provide for faster storage of data structures in the system memory 126. Additional EDC components can be employed for additional adjacent bit pair domains.

The system memory 126 is comprised of a plurality of memory devices 128, labeled memory device #1 through #K, where K is an integer greater than one. The system memory 126 is coupled to a data bus 146 and an address bus 148. The size of the data bus 146 and data structure is equal to the number of memory devices 128

5   multiplied by the bit column width of the memory devices 128. For example, the bit column width can be 4 with K being 72 to provide a 288-bit data structure. The data separator device 124 is coupled to the system memory 126 *via* the data bus 146. The memory controller 122 is coupled to a crossbar device (not shown) over a crossbar bus 142 and a mid-bus 144. The crossbar device can be coupled to a plurality of

10   microprocessor devices, input/output devices and/or memory devices (*e.g.*, one or more cache memory devices). The mid-bus 144 connects the memory controller 122 to the data separator device 124.

During a read operation, a data structure associated with an address selected *via* the address bus 148 is read from the plurality of memory devices 128 and

15   provided to the data separator 124 *via* the data bus 146. The data separator 124 separates or partitions the data structure into a first adjacent bit pair domain and a second adjacent bit pair domain. The first adjacent bit pair domain includes a first set of adjacent bit pairs from each of the plurality of memory devices associated with a given row address, and the second adjacent bit pair domain includes a second set of

20   adjacent bit pairs from each of the plurality of memory devices 128 associated with the same given row address, such that neither the first adjacent bit pair domain nor the second adjacent bit pair domain contain more than one adjacent bit pair from any single memory device.

The data bits associated with the first adjacent bit pair domain are transmitted

25   from the data separator 124 to a multiplexer 130 over the mid-bus 144. The data bits associated with the first adjacent bit pair domain can be transmitted in full or partitioned and transmitted in portions and recombined by the multiplexer 130. The multiplexer 130 transmits the first adjacent bit pair domain to the first EDC component 132, while the data separator 124 is transmitting the data bits associated

30   with the second adjacent bit pair domain. The data bits associated with the second adjacent bit pair domain can be transmitted in full or partitioned and transmitted in portions and recombined by the multiplexer 130. The multiplexer 130 transmits the second adjacent bit pair domain data bits to the second EDC component 134, while the adjacent bit pair data separator 124 is transmitting data bits associated with the

first adjacent bit pair domain for a subsequently read data structure. In this manner, a processing pipeline is established, until all of the desired data structures are read from the system memory 126. After processing by the EDC components 132 and 134, the corrected data is transmitted to a data organizer 136 for aggregation and

5   reorganization to be transmitted to the crossbar device and routed to a requestor.

In view of the foregoing structural and functional features described above, methodologies in accordance with an aspect of the present invention will be better appreciated with reference to FIGS. 6-7. While, for purposes of simplicity of explanation, the methodologies of FIGS. 6-7 are shown and described as being

10   implemented serially, it is to be understood and appreciated that the present invention is not limited to the illustrated order, as some aspects could, in accordance with the present invention, occur in different orders and/or concurrently with other aspects from that shown and described. Moreover, not all illustrated features may be required to implement a methodology in accordance with an aspect of the present invention. It

15   is to be further understood that the following methodologies can be implemented in hardware, software (e.g., computer executable instructions), or any combination thereof.

FIG. 6 illustrates a methodology for storing data in accordance with an aspect of the present invention. The methodology begins at 200 where a data block is

20   received for storage in system memory. At 210, adjacent bit pairs are assigned to separate adjacent bit pair domains, such that a given adjacent bit pair domain contains a single adjacent bit pair associated with each memory device for a given address. The number of adjacent bit pair domains is based on the column widths of the memory devices forming the system memory. That is the number of domains is

25   equal to the number of adjacent bit pairs in a row of memory. At 220, check bits are generated associated with a respective adjacent bit pair domain. The check bits are then aggregated (e.g., filled in at an assigned location) with the data bits associated with a respective adjacent bit pair domain at 230. At 240, data associated with the adjacent bit pair domains are buffered in preparation for storage.

30   It is to be appreciated that the adjacent bit pair domains can be buffered in a sequential manner, such that one adjacent bit pair domain is being stored, while check bits are generated for a subsequent domain in a pipeline processing manner to facilitate storage speed of the received data block. It is also to be appreciated that the check bit generation can occur in parallel for separate adjacent bit pair domains to

facilitate storage speed of the received data block. The methodology then proceeds to 250.

At 250, the data associated with the adjacent bit pair domains are aggregated to provide a data structure for storage. At 260, the data structure is written into memory to corresponding columns of the plurality of memory devices, such that only one adjacent bit pair is stored in a memory device column for a given row address associated with a respective adjacent bit pair domain.

FIG. 7 illustrates a methodology for reading data in accordance with an aspect of the present invention. The methodology begins at 300 where a data structure is read from the system memory. At 310, the data structure is partitioned into separate adjacent bit pair domains, such that a single adjacent bit pair domain contains one adjacent bit pair from each memory device. The number of adjacent bit pair domains depends on the column widths of the memory devices forming the system memory. At 320, the bits associated with the respective adjacent bit pair domains are transmitted to one or more EDC components for error detection and correction processing in a sequential and/or parallel manner. A single EDC component can be employed for sequential processing, while multiple EDC components can be employed for parallel processing. The methodology then proceeds to 330.

At 330, an EDC technique is performed on the bits associated with a respective adjacent bit pair domain for each adjacent bit pair domain. The EDC technique determines the existence of errors in a adjacent bit pair domain by separating data bits from the check bits, performing an ECC routine on the data structure portion to generate a syndrome to produce a mask, and performing an exclusive-OR on the data structure portion and the mask to detect single bit errors and adjacent double bit errors. The EDC technique then corrects the single bit errors and adjacent double bit errors. The corrected data structure portions associated with respective adjacent bit pair domains are then buffered (*e.g.,* in sequence, in parallel) at 340. At 350, the data associated with each of the adjacent bit pair domain is aggregated with the check bits discarded to provide a data block. At 360, the data block is transmitted to a requestor (*e.g.,* a processor through a crossbar device).

What have been described above are examples of the present invention. It is, of course, not possible to describe every conceivable combination of components or methodologies for purposes of describing the present invention, but one of ordinary

13

skill in the art will recognize that many further combinations and permutations of the present invention are possible. Accordingly, the present invention is intended to embrace all such alterations, modifications and variations that fall within the spirit and scope of the appended claims.

5